# Project 4

lessons:

* Lesson 1 - Managing State
* Lesson 2 - UI + Redux
* Lesson 3 - Redux Middleware
* Lesson 4 - Redux with React
* Lesson 5 - Asynchronous Redux
* Lesson 6 - react-redux
* Lesson 7 - Real World Redux

Lesson 1 - Managing State

You’ll learn techniques to make your state more predictable by moving your state to a central location and establishing strict rules for getting, listening, and updating that state.

Lesson 2 - UI + Redux

You’ll learn to move away from having state live in the DOM by creating a vanilla JavaScript application whose state is managed entirely by Redux.

Lesson 3 - Redux Middleware

You’ll learn to create custom middleware and add it to your Redux store. This middleware will allow you to enhance your store by hooking into and intercepting actions before they reach any reducers.

Lesson 4 - Redux with React

You’ll learn how to add React to a Redux application and have the state of that application be managed by Redux.

Lesson 5 - Asynchronous Redux

You’ll learn to better abstract asynchronous events by creating your own custom Thunk middleware and adding it to your store.

Lesson 6 - react-redux

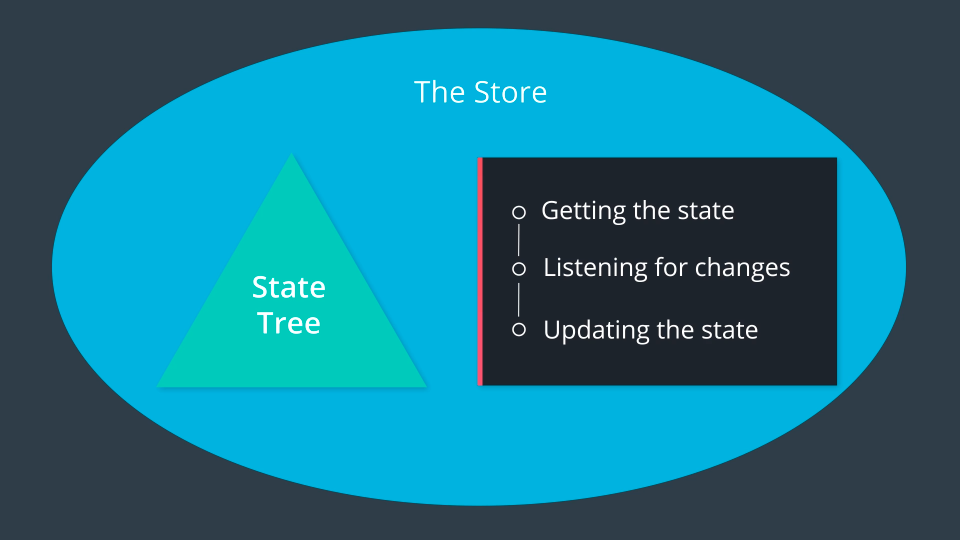
You’ll learn to leverage the react-redux bindings in order to leverage the benefits of a UI library like React and a state management library like Redux.

Lesson 7 - Real World Redux

You’ll take your knowledge of Redux and utilize it by building a real world Redux application. You’ll also learn advanced Redux topics like reducer composition and normalization.

In this section, we'll be building the store. If you remember from the previous section, the store has the following information:

* the state tree
* a way to get the state tree
* a way to listen and respond to the state changing
* a way to update the state

[[](https://classroom.udacity.com/nanodegrees/nd019/parts/7dab5516-d1ae-45d3-b8f8-d782b5534caf/modules/221d27be-a830-49a3-9803-9aa4a114489c/lessons/5b8c33c7-29d0-4fa4-9a03-ba49d1a2cb35/concepts/d137f3e4-6ddc-4c1f-b6e8-84fecaa19748)](https://classroom.udacity.com/nanodegrees/nd019/parts/7dab5516-d1ae-45d3-b8f8-d782b5534caf/modules/221d27be-a830-49a3-9803-9aa4a114489c/lessons/5b8c33c7-29d0-4fa4-9a03-ba49d1a2cb35/concepts/d137f3e4-6ddc-4c1f-b6e8-84fecaa19748)

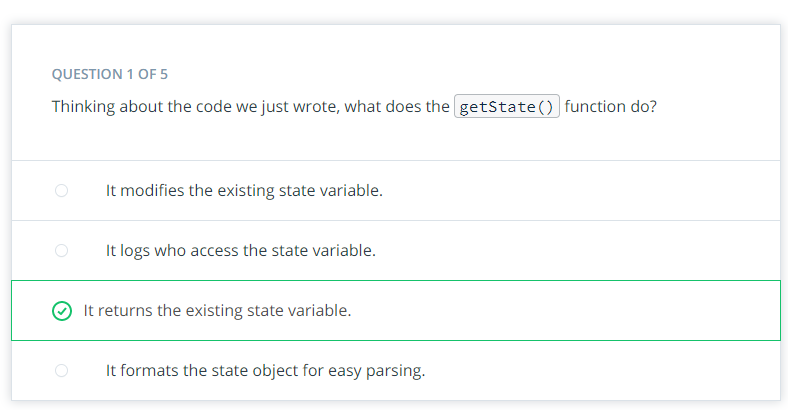
[The Store contains the state tree and provides ways to interact with the state tree.](https://classroom.udacity.com/nanodegrees/nd019/parts/7dab5516-d1ae-45d3-b8f8-d782b5534caf/modules/221d27be-a830-49a3-9803-9aa4a114489c/lessons/5b8c33c7-29d0-4fa4-9a03-ba49d1a2cb35/concepts/d137f3e4-6ddc-4c1f-b6e8-84fecaa19748)

So this is what we're going to do in this lesson - we're going to actually *create* the store code ourselves, from scratch.

<https://www.youtube.com/watch?time_continue=1&v=YqmnAPNCxkQ>

In this screencast, we started building out the createStore function. Currently, this factory function:

* takes in no arguments
* sets up a local (private) variable to hold the state
* sets up a getState() function
* returns an object that publicly exposes the getState() function



Our list of things we need to build for the store is shrinking:

* ~~the state tree~~
* ~~a way to get the state tree~~
* a way to listen and respond to the state changing
* a way to update the state

Our next task on the list is to make a way to listen for changes to the state.

<https://www.youtube.com/watch?v=AWOuF_qoEh8>

<https://www.youtube.com/watch?v=5jVn0L7nlBA>